IN THE SPECIFICATION

Please amend Page 5 of the Specification by adding the following paragraphs prior to the paragraph beginning "The cluster deployment flow (CDF) in one embodiment includes . . . " at line 24:

As disclosed in U.S. Patent Application Serial Number 10/378,503, which has been incorporated herein by reference in its entirety, the network deployment server 112 of the present disclosure includes a deployment tool that uses a plug-in architecture to isolate and modularize application specific functionalities and properties unique to each application server 102-110. Examples of the unique functionalities and properties may include application server specific security information and factors such as whether caching is to be performed. Other application server specific criteria include a particular location of an application server where the EAR file needs to be stored, whether an EAR file needs to be pre-processed to include application server required classes. Further, many application servers generally require that application server specific deployment descriptors be used.

Accordingly, application specific deployment configuration and functionalities for a given target application server 102-110 are encapsulated into a plug-in module 134 or that target application. The plug-in modules 134 implement a plug-in deployer interfaces and other helper interfaces to effect communication between the classes that implement the deployment wizard interface.

In one embodiment, the deployment tool 112, requests a server profile selection to determine which server profile is to be used to deploy a specified package of components or applications such as the archive J2EE application (EAR) file. For example, one or more plug-ins that are currently installed are determined and used to form a list of available target servers. That is, if an IBM Websphere plug-in is installed, then the Server Profile Manager deployment tool returns IBM Websphere application server as one of the candidate target servers.

In one embodiment, a server profile panel is presented to a user to select a target server from a list of target servers determined as described above. A user may select one from the list or may specify a new target application server 102-110, creating a new server profile and installing a plug-in 134 associated with the new server.

A plug-in 134 refers to an accessory program that enhances a main application. Plug-ins 134 are program units that may be added to the main application without affecting the main application. Because the deployment tool 112 of the present disclosure uses a plug-in architecture for application specific functionalities, additional target application servers 102-110 may be added easily by, for example, adding plug-ins 134 for those new target application servers 102-110. Accordingly, new application servers 102-110 for deployment may be added dynamically without having to change or modify other parts of the deployment tool 112.

Typically, a deployer, that is, a person performing the deployment selects a server type and defines a server profile that is used to save deployment settings. Server profile may already have been created previously, for example, from a previous deployment session to the same application server. Server profiles are created for servers that have corresponding plug-ins 134 that implement the deployer interface. For example, server profiles are created and saved by the server profile manager and are populated with the information provided by the user during the deployment process. Server profiles store information needed by the plug-in 134, including such information as the profile name, host name, and the port number. Other information in a server profile may include the type of deployment platform, file transfer protocol ("ftp") user identifier ("id"), ftp password, and deployment directory. Additional information needed for a particular application server 102-110 may be requested from the user during the deployment process and stored in the server profile.

Once a plug-in implements the deployer interface, the deployment tool 112 may determine which platforms are supported by the plug-in 134 by invoking an instance of the plug-in method that implements the deployer interface. For example, a class in the plug-in implementing an interface provided for communicating platform information to the deployment tool class, for instance, the MultiPlatformPlugin interface defining getplatforms method, may provide a list of the platforms supported by that particular plug-in. After a platform is selected as shown at, a corresponding plug-in is activated to package the EAR file.

A deployer helper interface may be used by the plug-ins 134 and provides access to one or more methods for requesting services of the deployment tool classes implementing the deployment wizard interface. For instance, after the EAR file is packaged, a user may edit the deployment descriptors using the deployer helper interface. The deployer helper interface is passed to a plug-in, when the plug-in is activated by the deployment tool. Prior to displaying a summary page for a packaged EAR file, the deployment tool may optionally display the contents of the packaged EAR file to be modified. The modifications may be performed using the helper interface dialogs or by using a provided XML editor.

The packaged EAR file is then deployed to the selected target application server 102-110 using the deploy method implemented by the plug-ins 134. For instance, the deploy method may be invoked by the deployment tool 112. Although EJB.ear may include components that are being packaged and deployed, the deployment tool 112 of the present disclosure is enabled to handle a complete EAR file, which may contain both the EJB and web application components, and other files. Deployable components may be created by using any one of the available enterprise development environment tools. One such tool is Advantage Joe 3.0, which provides capability for modeling, building, and deploying components onto an application server. The disclosed deployment tool 112, in one embodiment, combines the Java archives (JAR) files and incorporates them into Enterprise archive (EAR) files, which may then be deployed to a target application server; regardless of which tools were used to create the JAR files and EAR files.

JAR files typically include one or more J2EE modules making up a J2EE application. A J2EE module is a collection of one or more J2EE components of the same component type such as web and EJB. Each J2EE module typically includes a corresponding deployment descriptor that contains declarative data required during the deployment of the components in the module.

A J2EE application includes one or more J2EE modules and one J2EE application deployment descriptor. J2EE application deployment descriptor generally describes the WAR and EJB JAR files and includes security and database information specific to the application, if any. A J2EE application is packaged using the Java archive (JAR) file

format into a file with ear filename extension. When composing a J2EE application, J2EE modules used in the application are selected, an application directory structure is created, J2EE module deployment descriptors are created, a deployment descriptor for the J2EE application is created, and the J2EE application is packaged.

Alternatively, a deployment tool of the sent disclosure, may receive input in the form of module selection and model information, for example, if the deployment tool 112 is being invoked as part of a development environment. That is, when operating as an integrated part of a development tool 112 such as Advantage Joe, the deployment tool 112, in one embodiment, is supplied with development tool's information model, a directory of J2EE modules packaged in JAR files, and a project selection.

A development tool's information model may include detailed information about the application such as description of the classes and internal logic and relationships between the classes. A project selection may be used to build and deploy project related objects, for example, by selecting from a list of objects including classes, specifications, methods, parameters, projects, and jars. Accordingly, a project selection allows the deployment tool 112 to locate the particular project output built using a development environment tool 112. This output may include an output directory containing classes resulting from the build process and also any applicable jar files created during the build process in the development environment tool. The deployment tool 112 in the present disclosure may use the project selection information to access the development tool's information model and the output directory in order to package the J2EE modules into an EAR file

In a particular embodiment, the EAR file may be expanded into its individual components so that one or more deployment descriptors in the EAR file may be extracted and modified. The deployer interfaces provided in the present disclosure may be used to modify the deployment descriptors. For instance, although the supplied or constructed EAR files contain deployment descriptors, these descriptors may need to be modified prior to the actual deployment to a target application server. As known to those skilled in the art, a deployment descriptor refers to an XML file provided for each module and application, and describes how the modules and applications are to be deployed.

In a particular embodiment, the target application server to which the EAR file is to be deployed may be selected. The selection may, for example, be determined by presenting the user with a list of available application servers that have corresponding plug-ins 134 and allowing the user to select an application server from the list. In addition, the user may be given an option to enter a new application server not listed in the list. Thereafter, a plug-in corresponding to the selected application server may be dynamically loaded. Any other plug-ins that are available may also be dynamically loaded at this time. A profile may then be created for the selected application server if one does not exist already. For example, a new application server may need a corresponding application server profile created. This application server profile includes information such as the host name and the port number of the target application server to which the EAR files is being deployed. A validity check may then be performed to make sure that the EAR file includes a valid version of a deployment descriptor and conforms to valid data type definitions (DTD). The deployment descriptors may be XML files and, therefore, may need to be associated with a valid document type declaration provided in a DTD. If the DTD needs to be edited, appropriate changes may be made to DTD.

For example, a user wanting to include any application server security tags for the deployment may edit the DTD file. The EAR file is repackaged. The repackaged EAR file may contain modified DTD FILE and descriptors. A target platform for the selected application server may then be selected. The selection may be determined, for example, by presenting a list of platforms that the selected application server runs on, and allowing the user to select a platform server from the list. The list of platforms supported by the selected application server is provided by the corresponding plug-in via, for example, the plug-in interface method implemented by the corresponding plug-in. Thereafter, the deployment process may begin. The deployment process may begin, for example, if a user presses a finish button after having selected a platform from the list of the platform presented to the user. In this case, the user pressing the finish button, or performing any analogous activity to indicate that all customizations pertaining to the selected application server are complete, triggers a deploy method implemented by the plug-in to be invoked.